
# »Лекционен курс

# »ООП1 (Java)

**Изрази**

# Регистрация

https://tinyurl.com/25lgtqke

# Променливи и стойности

- Променливи
  - В една програма се използват за съхраняване на данни (напр. цифри, букви, други символи, символни низове, ...)
  - Те могат да се разглеждат като един вид контейнери
- Стойности
  - Числото, буквата или друг елемент от данни в една променлива
  - Тази стойност може да се променя във времето
  - Във времето променливите могат да съдържат различни еднотипни стойности

# Добра практика

- Да избираме значещи имена на променливи
  - Имената да предполагат предвиденото използване на променливите или посочват вида данни, които те ще съхраняват
- Напр.,
  - Ако използваме променлива за да преброим нещо, можем да я наречем count
  - Ако използваме променлива за скоростта на автомобил, може да я наречем speed

*Идентификаторите в Java са сензитивни*

# Въвеждащ пример

Какво прави програмата?

*Изчислява броя яйца*

```
public class EggBasket {
    public static void main(String[] args) {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        numberOfBaskets = 10;
        eggsPerBasket = 6;
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println("Ако имате");
        System.out.println(eggsPerBasket + " яйца в кошница и ");
        System.out.println(numberOfBaskets + " кошници, тогава");
        System.out.println("общият брой на яйцата е " + totalEggs);
    }
}
```

# Въвеждащ пример

Какви променливи и стойности?

```
public class EggBasket {
    public static void main(String[] args) {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        numberOfBaskets = 10;
        eggsPerBasket = 6;
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println("Ако имате");
        System.out.println(eggsPerBasket + " яйца в кошница и ");
        System.out.println(numberOfBaskets + " кошници, тогава");
        System.out.println("общият брой на яйцата е " + totalEggs);
    }
}
```

# Въвеждащ пример

Какъв резултат?

```
Ако имате
6 яйца в кошница и
10 кошници, тогава
общият брой на яйцата е 60

Process finished with exit code 0
```

```
public class EggBasket {
    public static void main(String[] args) {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        numberOfBaskets = 10;
        eggsPerBasket = 6;
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println("Ако имате");
        System.out.println(eggsPerBasket + " яйца в кошница и ");
        System.out.println(numberOfBaskets + " кошници, тогава");
        System.out.println("общият брой на яйцата е " + totalEggs);
    }
}
```

# Втори пример

» Проверка за това дали определена стойност (between) лежи в даден интервал (min, max) можем да зададем като израза

```
boolean isBetween = between > min && between < max;
```

# Съставни елементи на изразите

- Могат да се образуват изрази, които съдържат:
  - Литерали
  - Променливи
  - Символни константи
  - Оператори
- Съществуват правила за приоритети на операторите
  - Аналогично като в математиката

# Извиквания на методи

- Изразите могат да съдържат извиквания на методи
  - Могат да се появяват в дясната част на операторите за присвояване.
    - + С актуални параметри, които могат да бъдат изрази.
  - Подобно на изразите те произвеждат стойности когато се обработват.

# Валидно присвояване?

- Java винаги знае типа на стойността на изразите
  - Внимание: те трябва да съответстват на типовете на променливите, в които се записват стойностите

```
int i;
double x;

i = 10.3 * x;
```

*Присвояването невалидно!*

# Конвертиране

- Внимание с конвертирането на изрази, съдържащи различни типове данни
  - Възможно е автоматично конвертиране
    - Когато не се губи информация
  - При загуба на информация
    - Автоматично конвертиране не се извършва
- Можем да предизвикаме конвертиране
  - Оператор cast

# Автоматично преобразуване на типове

- Автоматично преобразуване в “по-големия” тип:
  - Без загуба на информация

Колко трансформации?

```
int a;
float b;
double x;

x = a + b;
```

2

# Явно преобразуване

» В Java (и в повечето езици за програмиране) данните могат явно да се трансформират от един в друг тип (cast)

» Обикновено това се използва като обратното действие на автоматичното преобразуване

```
double distance = 9.0;
int points = distance;
```

Коментар?

```
int i;
double x;

i = 10.3 * x;
```

*Грешни присвоявания*

# Пример за явно конвертиране

```
int i;
double x;

i = (int)(10.3 * x);
```

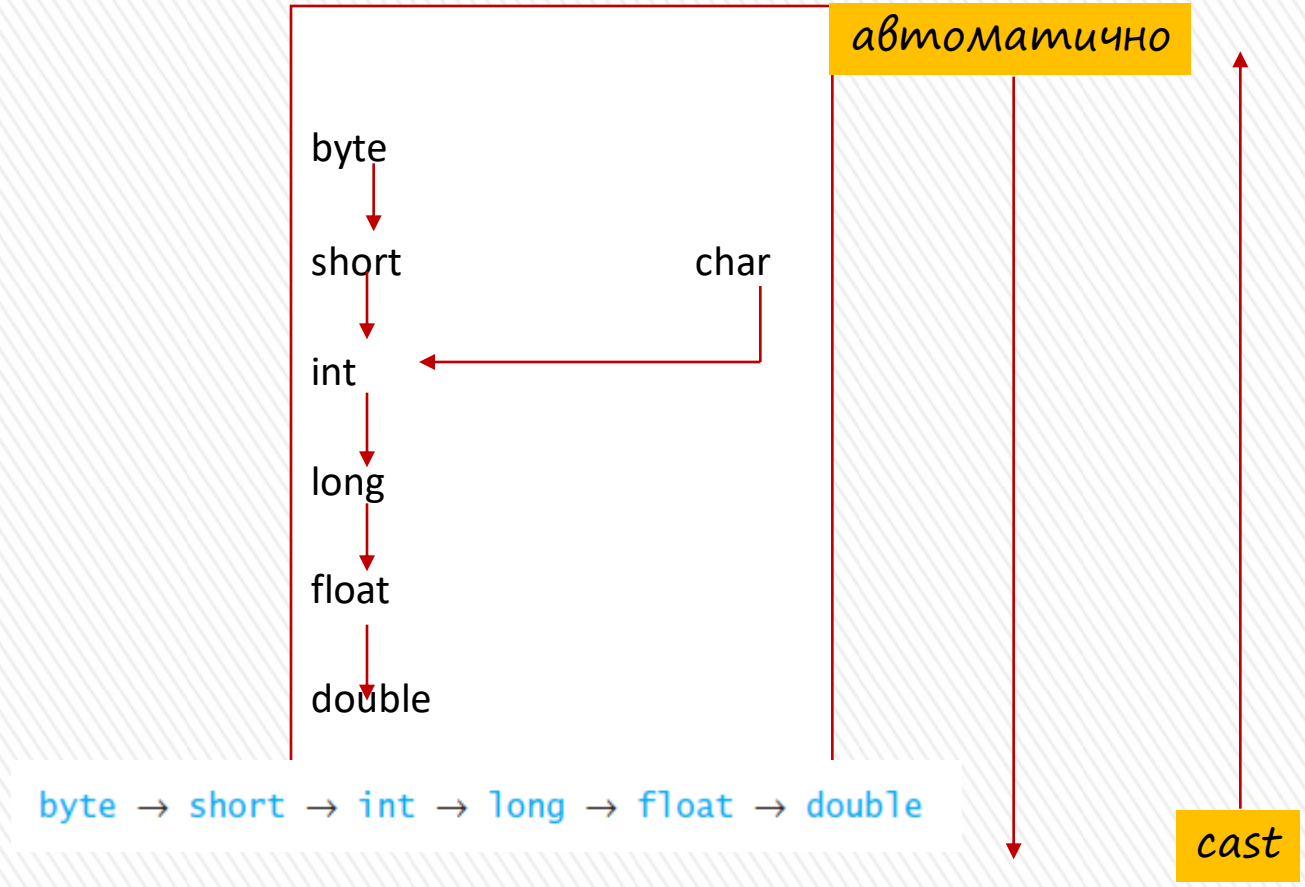
# Пример

```
1  public class ErrAssignments {
2      public static void main (String argv[]) {
3          double distance = 9.0;
4          int point = distance;
5
6          int i;
7          double x = 2.0;
8
9          i = 10.3 * x;
10
11         System.out.println("Стойност на i =" + i);
12     }
13
14 }
```

```
1  public class ErrAssignments {
2      public static void main (String argv[]) {
3          double distance = 9.0;
4          double point = distance;
5
6          int i;
7          double x = 2.0;
8
9          i = (int)(10.3 * x);
10
11         System.out.println("Стойност на i =" + i);
12     }
13
14 }
```

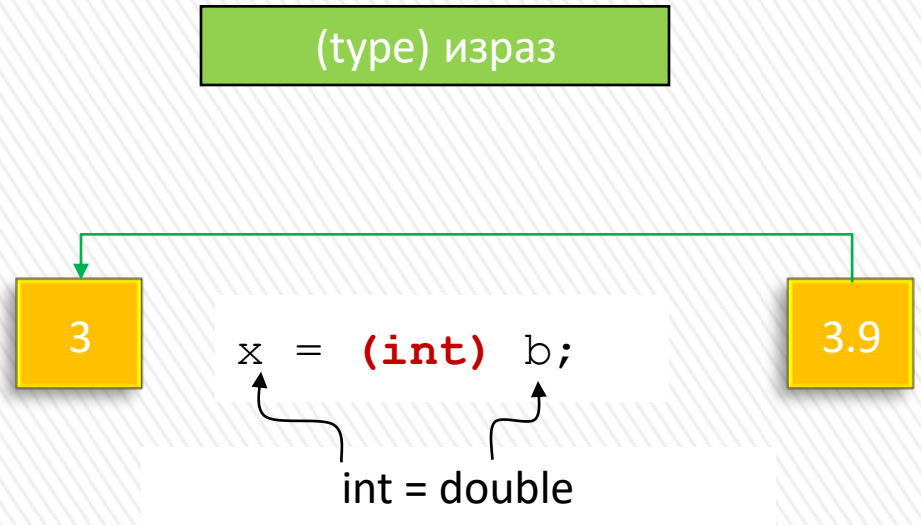
# Преобразуване на типове: cast-оператор

*Проблем:Операнди от различен тип* `x = y;  x = a + b;`



# Явно преобразуване на типове: type-cast–оператор

» Преобразува израза в нов израз от тип *type* - ев. със загуба на информация



# Пример

```
1  public class ErrAssignments {
2      public static void main (String argv[]) {
3          double distance = 9.0;
4          double point = distance;
5
6          int i;
7          double x = 2.0;
8
9          i = (int)(10.3 * x);
10
11         System.out.println("Стойност на i =" + i);
12     }
13
14 }
15
```

Properties ▪ Servers ▪ Data Source Explorer ▪ Snippets Problems Console

```
<terminated> ErrAssignments [Java Application] C:\Program Files\Java\jre1.8.0_341\bi
Стойност на i =20
```

# Три проблема с изразите

1. Притежават ли операндите коректните типове и свързани с тях операциии?
2. Коректно ли са свързани подизразите?
3. Ако подизразите са определили вече общата стойност: да бъде ли оценен остатъчният израз?

# Пример

**Какво прави изразът?**

```
((jahr % 4) == 0)
&& ((jahr % 100) != 0)
|| ((jahr % 400) == 0)
```

*Високосна година:*

- *Всички години, които се делят на 4, но не се делят на 100*
- *Всички години, които се делят на 400*

# Проблем 1

Какви операции?

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```

## Класове операции

Аритметични операции (modulo)

Логически операции (and, or)

Релационни операции (равно, различно)

# Проблем 2

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```

1. Приоритети на операции
2. Повече скоби?
3. По-малко скоби?

# Проблем 3

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```

- ((year % 4) == 0) *грешно* → . . . && . . . *грешен*
- ((year % 4) == 0) && ((year % 100) != 0) *вярно*
  → . . . || . . . *верен*

Java: &&, || със съкратено оценяване
& | с пълно оценяване

# Пример

```
import java.util.Scanner;
public class ChangeMaker {
    public static void main(String[] args) {
        int amount, originalAmount,
                s50, s10, s5, s1;
        System.out.println("Въведи стойност от 1 до 99.");
        System.out.println("Ще намеря комбинацията от монети,");
        System.out.println("която отговаря на стойността:");
        Scanner keyboard = new Scanner(System.in);
        amount = keyboard.nextInt();
        originalAmount = amount;
        s50 = amount/50;
        amount = amount%50;
        s10 = amount/10;
        amount = amount%10;
        s5 = amount/5;
        amount = amount%5;
        s1 = amount;
        System.out.println(originalAmount +
                " стотинки може да бъде дадена като:");
        System.out.println(s50 + " монети по 50");
        System.out.println(s10 + " монети по 10");
        System.out.println(s5 + " монети по 5 и");
        System.out.println(s1 + " монети по 1");
    }
}
```

Какво прави програмата?

*Въведана сума като монети*

# Пример

```
import java.util.Scanner;
public class ChangeMaker {
    public static void main(String[] args) {
        int amount, originalAmount,
                s50, s10, s5, s1;
        System.out.println("Въведи стойност от 1 до 99.");
        System.out.println("Ще намеря комбинацията от монети,");
        System.out.println("която отговаря на стойността:");
        Scanner keyboard = new Scanner(Syst
        amount = keyboard.nextInt();
        originalAmount = amount;
        s50 = amount/50;
        amount = amount%50;
        s10 = amount/10;
        amount = amount%10;
        s5 = amount/5;
        amount = amount%5;
        s1 = amount;
        System.out.println(originalAmount +
                " стотинки може да бъде дадена като:");
        System.out.println(s50 + " монети по 50");
        System.out.println(s10 + " монети по 10");
        System.out.println(s5 + " монети по 5 и");
        System.out.println(s1 + " монети по 1");
    }
}
```

Какъв резултат?

```
Въведи стойност от 1 до 99.
Ще намеря комбинацията от монети,
която отговаря на стойността:
99
99 стотинки може да бъде дадена като:
1 монети по 50
4 монети по 10
1 монети по 5 и
4 монети по 1

Process finished with exit code 0
```

# Синтаксис на изразите в Java: странични ефекти

- Изрази в Java
  - Литерали (числа, символни низове, логически стойности, ...)
  - Поменлива (+ параметър)
  - Извиквания на методи
  - Съставни изрази с аритметични, релационни, логически и побитови операции
  - Присвоявания (!!!)
  - ...

*Критика (към Java, C):*
*Няма ясно разделение между изрази и оператори !*

*→ Причина за грешки !*

# Пример

Крайни стойности?

```
class Assignment {

   public static void
                main (String[] args) {

        int x = 2, y = 5, z = 1;

        x = (z++ - (y = y + x));

   }

}
```

X = -6 Y = 7 Z = 2

Избягване : странични ефекти в изрази!
(заедно с изчисляването на стойност: промяна на стойностите на променливите – причина за грешки)

# Странични ефекти: z++ и ++z

```
x = ( z++ - (y = y + x));
```

Стойност = z
След това: z = z + 1
Резултат: x = -6, y=7, z=2

1-7

```
x = ( ++z - (y = y + x));
```

първо: z = z + 1
стойност = z + 1
резултат: x = -5, y=7, z=2

2-7

# Странични ефекти: принципно избягвани

*Но: съществуват смислени приложни случаи !*

*Задача на един израз:*
- *Да:изчисляване на стойности*
- *Не:промяна на стойности*

```
x = (z++ - (y = y + x));
```

```
y = y + x;
x = z - y;
z++;
```

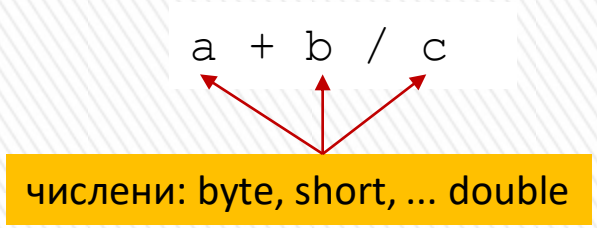
*Експлицитно задаване на стойностите на променливите, които трябва да се променят (последователност!)*

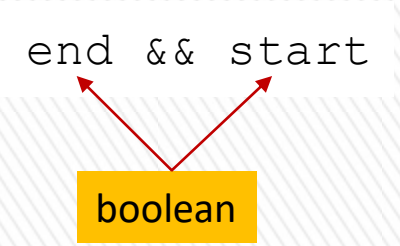
*По-добре:*
z = z +1;

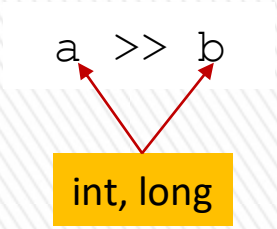
```
x = (++z - (y = y + x));
```

```
++z;
y = y + x;
x = z - y;
```

# Операции: свързани с типове



# Обобщение: примитивни типове данни

| Тип | Дължина (Byte) | Област на стойности |
|---|---|---|
| boolean | 1 | true, false |
| char | 2 | Всички Unicode-символи |
| byte | 1 | $-2^7 \ldots 2^7 - 1$ |
| short | 2 | $-2^{15} \ldots 2^{15} - 1$ |
| int | 4 | $-2^{31} \ldots 2^{31} - 1$ |
| long | 8 | $-2^{63} \ldots 2^{63} - 1$ |
| float | 4 | $+ / - 3.4028234738 * 10^{38}$ |
| double | 8 | $+ / - 1.79769313486231570 * 10^{308}$ |

# Аритметични операции

- Числови операнди,  тип на резултат: числов
- Конвертиране в обхващащия тип при операнди с различни типове

| | | |
|---|---|---|
| + | Positive | n |
| - | Negative | –n |
| + | Sum | a + b |
| - | Difference | a – b |
| * | Product | a * b |
| / | Quotient | a / b |
| % | Restvalue | a modulo b |
| ++ | Preincrement | ++a става a+1, увеличава a с 1 |
| ++ | Postincrement | a++ става a, увеличава a с 1 |
| -- | Predecrement | ––a става a-1, намалява a с 1 |
| -- | Postdecrement | ––a става a, намалява a с 1 |

Забележка: % също за числа с плаваща запетая !

# Операции: претоварване (Overloading)

*x = a + b*

```
+:  int    x  int    →  int
+:  float  x  float     →  float
+:  double x  ...
```

*"Overloading": еднакво име за различни операции*

# Операции: изискват определено позициониране на операндите

Infix–операция:

```
a + b
a << b
```

Postfix–операция:

```
a ++
```

Prefix–операция:

```
++ a
! true
~ a
^ a
```

Други (3-позиционни):

```
a > b ? a : b
```

3 – позиционна Infix-операция

# Релационни операции

- За числени операнди (също смесени)
- (Не) Равенство също за типове на обекти (сравняване на адреси)
- Тип на резултата: boolean

| | | |
|---|---|---|
| == | равно | a == b |
| != | неравно | a != b |
| < | по-малко | a < b |
| <= | по-малко равно | a ≤ b |
| > | по-голямо | a > b |
| >= | по-голямо равно | a ≥ b |

# Тип 'boolean'

» Стойности: true, false

```
boolean ready, ...
ready = false;
ready = jahr > 1999;
```

Операции:

| | |
|---|---|
| ! | *Отрицание* |
| & | *Конюнкция (‘and’)* |
| \| | *Дизюнкция (‘or’)* |

*отговарят на тези в логиката*

# Логически операции

- За логически типове (boolean)
- Тип на реултата: boolean
- „Частично оценяване": следващите, в дясно стоящи подизрази не се оценяват, ако стойността вече е известна
  - напр. a && b → false, ако a е вече грешен

    → b не се оценява

| | | |
|---|---|---|
| ! | отрицание | ~a |
| && | AND с частично оценяване | $a \wedge b$ |
| \|\| | OR с частично оценяване | $a \vee b$ |
| & | AND с пълно оценяване | $a \wedge b$ |
| \| | OR с пълно оценяване | $a \vee b$ |
| ^ | EXCLUSIVE-OR (или … или) | $a \otimes b$ |

# Побитови операции

» Манипулации с битове за `int` съотв. `long`

| | | |
|---|---|---|
| ~ | Единичен комплимент | ~a: инвертиране битовете на a |
| \| | Побитов OR | a \| b: побитов $a_i \vee b_i$ |
| & | Побитов AND | a & b: побитов $a_i \wedge b_i$ |
| ^ | Побитов XOR | a^b: побитов $a_i \otimes b_i$ |
| >> | Писане дясно със знак | a >> b: битовете на a с b позиции надясно, знак като при a |
| >>> | Писане дясно без знак | a >>> b: битовете на a с b позиции надясно, попълване с 0, надписване на знак (0) |
| << | Писане ляво | a << b: битовете на a с b позиции наляво, попълване с 0, знак като при a |

При това: b modulo 32 (`int`) съотв. 64 (`long`)

| | |
|---|---|
| a << 1 | Умножение с 2 |
| a << 2 | Умножение с 4 |
| a << n | Умножение с $2^n$ |

# Оператори за присвояване

- Присвояванията за изрази във формата

  *ЛяваСтрана* **оператор_за_просвояване** *Израз*

  - напр.: **`x = x + y`**
- ЛяваСтрана означава памет (в общия случай променлива)

Разлика за "x" в **`x = x + y`**;


- Тип на присвояването = Тип на ЛяватаСтрана
- Стойност на присвояването = стойност на израза

*Смесване "=" / "==", (само) при boolean:*

```
boolean x = true, y = false;
System.out.println(y == x);
System.out.println(y = x);
```

# Оператори за присвояване

EBNF:

Оператор за присвояване ::= = | += | *= | –= | /=
| %= | &= | ^=
| <<= | >>= | >>>=.

*Комбинация на "=" с аритметични и побитови операции във формата "Операция=" за операциите*

+ | * | - | / | % | & | ^ | << | >> | >>>

*Ефект за x операция= y както x = x операция y*

| | | |
|---|---|---|
| x += 100 | *както* | x = x + 100 |
| x <<= 2 | *както* | x = x << 2 |

# Други оператори

? оператор:

ЛогическиИзраз ? израз1 : израз2

доставя:

израз1, ако ЛогичекиИзраз == true
израз2, ако ЛогичекиИзраз == false
напр. x > y ? max = x : max = y

*Свързване на низове:*

За символни низове (Strings): string1 + string2

```
x = 1; y = 2;
System.out.println(x + y);  → Изход: "3"
System.out.println(x + "" + y); → Изход: "12"
```

# Други оператори

new – оператор:

*Създаване на инстанции:* new Typ ([ArgumentList])

С конструктура *Typ([Argumentlist])*

За инициализиране на една инстанция

instanceof – оператор: InstanceName instanceof ClassName

*Тип на резултата* ***boolean****:*

***true****, ако InstanceName е инстанция на класа ClassName,*
*съотв. означава един подклас на ClassName*

# Преглед на операциите:приоритети

| Операция | Типизиране | Асоциативност | Означение |
|---|---|---|---|
| | Група 1 | | |
| ++ | N | R | Increment |
| -- | N | R | Decrement |
| + | N | R | Uneres Plus |
| - | N | R | Uneres Minus |
| ~ | I | R | Onecomplement |
| ! | L | R | Negation |
| (type) | A | R | Type-Cast |
| | Група 2 | | |
| * | N,N | L | Multiplication |
| / | N,N | L | Division |
| % | N,N | L | Modulo |
| | Група 3 | | |
| + | N,N | L | Addition |
| - | N,N | L | Extraction |
| + | S,S | L | String concatenation |
| | Група 4 | | |
| << | I,I | L | Left write |
| >> | I,I | L | Right write |
| >>> | I,I | L | Right write with Nullexpansion |

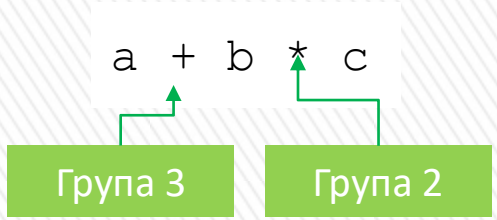
| Операция | Типизиране | Асоциативност | Означение |
|---|---|---|---|
| | | Група 5 | |
| < | N,N | L | Smaller |
| <= | N,N | L | Smaller eq. |
| > | N,N | L | Bigger |
| >= | N,N | L | Bigger eq. |
| instanceof | R,R | L | Class membership |
| | | Група 6 | |
| == | P,P | L | Equally |
| != | P,P | L | Unequally |
| == | R,R | L | Referece eq. |
| != | R,R | L | Reference uneq. |
| | | Група 7 | |
| & | I,I | L | Bitw. AND |
| & | L,L | L | Log. AND |
| | | Група 8 | |
| ^ | I,I | L | Bitw. XOR |
| ^ | L,L | L | Log. XOR |
| | | Група 9 | |
| \| | I,I | L | Bitw. OR |
| \| | L,L | L | Log. OR |
| | | Група 10 | |
| && | L,L | L | Log. AND, Short-Cut |

| Операция | Типизиране | Асоциативност | Означение |
|---|---|---|---|
| | | Група 11 | |
| \|\| | L,L | L | Log. OR, Short-Cut |
| | | Група 12 | |
| ? : | L,A,A | R | Conditional evaluation |
| | | Група 13 | |
| = | V,A | R | Assignment |
| += | V,N | R | Addition assignment |
| -= | V,N | R | Extraction assignment |
| *= | V,N | R | Multiplication ssignment |
| /= | V,N | R | Division assignment |
| %= | V,N | R | Rest value assignment |
| &= | V,N | R | Bitw.-AND-Assignment |
| | V,L | R | Log.-AND-Assignment |
| \|= | V,N | R | Bitw.-OR-Assignment |
| | V,L | R | Log.-OR-Assignment |
| ^= | V,N | R | Bitw.-XOR-Assignment |
| | V,L | R | Log.-XOR-Assignment |
| <<= | V,I | R | Left-Write-Assignment |
| >>= | V,I | R | Right-Write-Assignment |
| >>>= | V,I | R | Right-Write-Assignment with Nullexpansion |

# Правила за приоритети

Приоритет = Сила на свързване

- Таблица: по-ниска група с по-висок приоритет




*Еквивалентен на:* `a + ( b * c )`

*Сравнение:* `(int) 3.1 + 3.9` *и* `(int) (3.1 + 3.9)`

# Правила за приоритети

- Вътре в една група: приоритет по асоциативност
  - Напр. лява асоциативност L:

```
a - b - c        както        ( a - b ) - c
```

```
a - ( b - c ) би било грешно
```

```
a == b == c      както        ( a == b ) == c
```

true/false

Тип на a, b, c:
int възможен ?

# Правила за приоритети

- Вътре в една група: приоритет по асоциативност
  - напр. дясна асоциативност R:

```
a = b = c
```

*както*

```
a = ( b = c )
```

*след това: a, b, c със стойност на c*

*избягване ?!*

# Типизиране

( **N** ) Числен
( **I** ) Интегрален
( **L** ) Логически
( **S** ) Низ
( **R** ) Референция
( **P** ) Примитивен
( **A** ) Всички типове
(Типове на операндите)
( **V** ) Лява страна на присвояване

# Внимание при релационни оператори

```
6  class Number {
7      int i;
8  }
9
10 public class Assignment {
11     public static void main(String[] args) {
12         Number n1 = new Number();
13         Number n2 = new Number();
14         n1.i = 9;
15         n2.i = 47;
16         System.out.println("1: n1.i: "+ n1.i + ", n2.i: " + n2.i);
17         n1 = n2;
18         System.out.println("2: n1.i: "+ n1.i + ", n2.i: " + n2.i);
19         n1.i = 27;
20         if (0.1 + 0.2 == 0.3)
21             System.out.println("3: n1.i: "+ n1.i + ", n2.i: " + n2.i);
22         if (0.25 + 0.25 == 0.5)
23             System.out.println("4: n1.i: "+ n1.i + ", n2.i: " + n2.i);
24         if (1 + 2 == 3)
25             System.out.println("5: n1.i: "+ n1.i + ", n2.i: " + n2.i);
26     }
27 }
```

Properties ▪ Servers ▪ Data Source Explo

<terminated> Assignment (2) [Java Applica

```
1: n1.i: 9, n2.i: 47
2: n1.i: 47, n2.i: 47
4: n1.i: 27, n2.i: 27
5: n1.i: 27, n2.i: 27
```

# Ключова дума var

```
public class VarExample {
    public static void main(String[] args) {
        var a = 10;
        var b = 20;
        System.out.println(a+b);
    }

}
```

**Дефиниция и употреба**:
Ключовата дума var позволява инициализирането на променлива, без да е необходимо да се декларира нейният тип. Типът на променливата зависи от типа на данните, които ѝ се присвояват.
Ключовата дума var е въведена в Java 10.


Благодаря за вниманието!